The specified requirements for the assignment are as follows:

1.Use PHP to reuse common elements in your website, such as menu, header, and footer.
- The first specified requirement for the assignment is to use PHP to reuse common elements in your website. This means that you should rewrite your web pages so that common static HTML elements such as menu, header, and footer are written in common text files that are then “included” back into your web pages using PHP ‘include’ statements. This allows you to modularize and reuse your web application code, making it easier to maintain and update.

2.Create an ‘orders’ database table in your MySQL database to record data sent from the “payment.php” form.
- The second specified requirement for the assignment is to create an ‘orders’ database table in your MySQL database. This table will record the data sent from the “payment.php” form, including information on the customer, product, and payment details. The format of the database fields should match appropriate validation rules defined in previous assignments. In addition to the fields that record information from the “payment” form, you should also add fields for an auto-generated primary key called ‘order_id’, the total cost of the order ‘order_cost’, date/time of order ‘order_time’, and a field called ‘order_status’ that can have one of three values: PENDING, FULFILLED, PAID, or ARCHIVED.

3.Create a file called “settings.php” to store your database connection variables and use it to connect to your MySQL database.
- The third specified requirement for the assignment is to create a file called “settings.php” to store your database connection variables. This file should contain the connection variables needed to connect to your MySQL database on the feenix-mariadb database server, and should be used in your PHP code to connect to the database. Using a separate file for your database connection variables makes it easier to port your website to different environments, such as from a development environment to a testing environment.

4.Extend the enquire form to enable a user to purchase a product/service by adding price details, quantity fields, and credit card payment details.
-4.1
+ The fourth specified requirement, sub-section 4.1, is to extend the enquire form to enable a user to purchase a product/service by adding price details, quantity fields, and credit card payment details. This can be done by either renaming the enquire.php file to payment.php and adding the necessary inputs or by creating a new payment.php file and making the enquire page submit to it. You will need to add price details for various items in the product/service range, add fields for selecting the quantity of the product/service to be purchased, and add form inputs for credit card payment details, including credit card type, name on credit card, credit card number, credit card expiry date, and card verification value (CVV). The form should have a ‘submit’ button with the caption ‘Check Out’ that submits the form to a “process_order.php” page.
-4.2

+ The fourth specified requirement, sub-section 4.2, is to disable HTML validation in the payment form. While client-side validation using HTML5 was used in previous assignments, server-side data format checking should be implemented in this assignment to preserve the integrity of the server data. To test that server-side validation works correctly, you should add the novalidate attribute to your forms to disable client-side HTML5 validation.

5.Process orders using a “process_order.php” page that sanitizes and validates all values received from the “payment.php” form before storing the order in the orders table.
- The fifth specified requirement for the assignment is to process orders using a “process_order.php” page. This page should receive data from the “payment.php” form, sanitize all values to remove leading and trailing spaces, backslashes, and HTML control characters, and validate the data format according to specified rules before storing the order in the orders table. If the input data does not meet the format requirements, errors should be returned to a “fix_order.php” form that displays all form control fields filled with data entered in the enquire and payment pages, with errors marked or highlighted. If the input data is correctly validated, the total cost of the order should be calculated, the order should be stored in the orders table, and an order receipt webpage called “receipt.php” should be returned to the user.
6.Create a manager’s order report and order update page called “manager.php” that allows the manager to make queries about orders, display the result in an HTML table, and update the status of an order.
- The sixth specified requirement for the assignment is to create a manager’s order report and order update page called “manager.php”. This page should allow the manager to make queries about orders, display the result in an HTML table, and update the status of an order. The manager should be able to display all orders, orders for a specific customer based on their name, orders for a particular product, orders that are pending, and orders sorted by total cost. The manager should also be able to update the status of an order from PENDING to FULFILLED, PAID, or ARCHIVED, and cancel (i.e., delete) a pending order.

Checklist:

1.Use PHP to reuse common elements in your website:

-Rewrite web pages so that common static HTML elements such as menu, header, and footer are written in common text files.
-Use PHP ‘include’ statements to include these common elements back into your web pages.
2.Create an ‘orders’ database table in your MySQL database:

-Create an ‘orders’ table to record data sent from the “payment.php” form.
-Add fields for an auto-generated primary key called ‘order_id’, the total cost of the order ‘order_cost’, date/time of order ‘order_time’, and a field called ‘order_status’ that can have one of three values: PENDING, FULFILLED, PAID, or ARCHIVED.
3.Create a file called “settings.php” to store your database connection variables:

-Create a file called “settings.php” that contains the connection variables needed to connect to your MySQL database on the feenix-mariadb database server.
-Use this file in your PHP code to connect to the database.
4.1. Extend the enquire form to enable a user to purchase a product/service:

-Add price details for various items in the product/service range.
-Add fields for selecting the quantity of the product/service to be purchased.
-Add form inputs for credit card payment details, including credit card type, name on credit card, credit card number, credit card expiry date, and card verification value (CVV).
-Add a ‘submit’ button with the caption ‘Check Out’ that submits the form to a “process_order.php” page.
4.2. Disable HTML validation in the payment form:

-Add the novalidate attribute to your forms to disable client-side HTML5 validation.
5.Process orders using a “process_order.php” page:

-Receive data from the “payment.php” form and sanitize all values.
-Validate the data format according to specified rules before storing the order in the orders table.
-If the input data does not meet the format requirements, return errors to a “fix_order.php” form.
-If the input data is correctly validated, calculate the total cost of the order, store the order in the orders table, and return an order receipt webpage called “receipt.php” to the user.
6.Create a manager’s order report and order update page called “manager.php”:

-Allow the manager to make queries about orders and display the result in an HTML table.
-Allow the manager to update the status of an order from PENDING to FULFILLED, PAID, or ARCHIVED.
-Allow the manager to cancel (i.e., delete) a pending order.

Enhancements Checklist:
-Create manager security:

+Create a “Manager registration” page with server-side validation requiring a unique username and a password rule.
+Store the manager’s username and password in a table.
+Create a “Manager Log-in” page to use the stored data and control access to the manager web pages.
+Ensure the manager web page cannot be entered directly using a URL.
+Create a “Manager Log-out” page.
+Provide a ‘log-out’ link on the manager page if ‘logged in’.
-Provide more advanced Manager reports based on compound queries:

+Create queries to display the most popular product ordered, fulfilled orders purchased between two dates entered by the vendor, or the average number of orders per day.

-On the table on the Manager page, provide the ability to select a column heading and re-sort the table:

+Add functionality to select a column heading and re-sort the table in the order of that field.
+If selected again, reverse the order.
-Store customers' details in a separate 'customers' data table:

+Create a 'customers' data table to store customers' details.
+Create a primary-foreign key link between the 'customers' and 'orders' tables.
-Store product details and options in a separate 'products' data table:

+Create a 'products' data table to store product details and options.
+Dynamically fill the product page with data from the 'products' data table.

Revised version of the Website Requirements Checklist to make it easier to read and understand:

-Use PHP to include common static elements such as menu, footer, and header.
-Create a "settings.php" file to store your database connection variables.
-Create an 'orders' table in your MySQL database.
-Disable HTML validation in the payment form.
-Process orders using a "process_order.php" page that sanitizes and validates all values received from the "payment.php" form before storing the order in the orders table.
-Create a manager's order report and order update page called "manager.php" that allows the manager to make queries about orders, display the result in an HTML table, and update the status of an order.
--Other deductions based on documentation, code, and file inspection:

-Do not use deprecated HTML elements/attributes.
-Use appropriate HTML semantics.
-Provide adequate code comments to inform later code understanding/maintenance.
-Ensure that all links are relative, not absolute, and are not broken.